

**Web Apps with Rust and Leptos**

**Course Number:** RUST-120
**Duration:** 3 days

**Overview**

Rust is an excellent language for building front-end web applications with WebAssembly. Leptos is a component-based front-end web framework similar to React, Angular, and Solid.js. In this Rust training course, attendees learn how to build front-end web applications with Rust and Leptos. Participants learn how to create components, pass data to components, and emit events from components. By the end of this course, students confidently build front-end web applications with Rust and Leptos.

**Prerequisites**

This course assumes prior experience with Rust, HTML, and CSS.

**Materials**

All students receive comprehensive courseware covering all topics in the course. Courseware is distributed via GitHub through documentation and extensive code samples.

**Software Needed on Each Student PC**

* A free, personal GitHub account to access the courseware
* Permission to install Rust and Visual Studio Code on their computers
* Permission to install Rust Crates and Visual Studio Extensions

If students cannot configure a local environment, a cloud-based environment can be provided.

**Objectives**

* Understand the principles of building a web application
* Understand WebAssembly and how it works
* Explore the differences between pure JS frameworks and Rust-based WebAssembly frameworks
* Create a user interface with components
* Incorporate best practices and principles of working with components
* Run server functions from a client-side code
* Publish and host a Leptos application

**Outline**

* Introduction
* WebAssembly with Leptos
	+ What is Web Assembly?
	+ What is Leptos?
	+ How does Leptos compare to React, Angular, and Blazor?
* Getting Started
	+ Create a Leptos Project
	+ Run and Debug a Leptos Project with Visual Studio Code
* Components
	+ What are Components?
	+ Create a Component
	+ Pass Data to a Component via Props
	+ Dynamic Attributes
	+ Passing Children to Components
* Parent-Child Components
	+ Communicate from Child to Parent via a Write Signal
	+ Communicate from Child to Parent via a Callback
	+ Use a Closure instead of a Callback
	+ Use an Event Listener
	+ Context
* Component Render Logic
	+ Display Data in a Component
	+ Expressions
	+ Conditionally Display Data in a Component
	+ Render a Collection of Data
	+ Error Handling
* Component Event Handling
	+ Handle Events in a Component
	+ Event Modifiers
	+ Event Propagation
* Reactivity with Signals
	+ What are Signals?
	+ Four Signal Operations
	+ Dependency between Signals
	+ Effects
* Router
	+ What is Routing?
	+ What is a Router?
	+ Define Routes
	+ Nested Routes
	+ Parameters and Queries
	+ Links and Forms
* Working with the Server
	+ Calling Server Functions
	+ Extractors
	+ Responses
	+ Redirects
* Forms
	+ What is an HTML Form?
	+ Collecting Data from Users
	+ Form Validation
	+ Processing Form Submissions
	+ Action Forms
* Metadata
* Styling
* Server-Side Rendering
	+ What is Server-Side Rendering?
	+ Using Cargo-Leptos
	+ Life Cycle of Loading a Leptos Page
	+ Rendering Modes
	+ Hydration
* Testing
	+ Unit Testing Components
	+ E2E Testing
* Deployment
	+ Build a Leptos Application
	+ Publish a Leptos Application
	+ Hosting Considerations
	+ Dockerize a Full-Stack Leptos Application
	+ Optimizing WASM Size
* Conclusion